U-Prove Range Proof Extension

Draft Revision 1

Microsoft Research

Author: Mira Belenkiy

June 2014

Summary

This document extends the U-Prove Cryptographic Specification [UPCS] by specifying set membership proofs. This allows proving that a committed value is less than, less than or equal to, greater than, or greater than or equal to another (committed) value.

Contents

Summary 1
1 Introduction 3
1.1 Notation 3
1.2 Feature overview 5
2 Protocol specification 5
2.1 Common Protocols 6
2.2 Presentation 8
2.3 Verification 15
3 Security Considerations 18
References 18

List of Figures

Figure 1: EQProofParams. 7
Figure 2: RangeProve 8
Figure 3: ComputeC 9
Figure 4: GetBitProofs. 10
Figure 5: GenerateBitDecomposition 11
Figure 6: DefaultBitDecomposition 11
Figure 7: ComputeD 12
Figure 8: ComputeX. 12
Figure 9: ComputeE 13
Figure 10: MainProof. 14
Figure 11: EqualityOfDL. 15

Change history

Version	Description
Revision 1	Initial draft

1 Introduction

This document extends the U-Prove Cryptographic Specification [UPCS] by specifying range proofs. The Prover will prove to the Verifier that a committed value is less than, less than or equal to, greater than, or greater than or equal to another (committed) value.

The Prover knows a secret value a, and will prove to the Verifier an inequality relation between a and another value b that may or may not be known to the Verifier. The Prover and Verifier have as common input a pair of generators $g, h \in G_q$. The Prover will create one of the following proofs:

$$\pi_{\odot} = PK\{\alpha, \beta, \gamma, \delta \big| C_A = g^{\propto} h^{\gamma} \cap\ C_B = g^{\beta} h^{\delta} \cap \propto \odot\ \beta\}$$

or

$$\pi_{\odot} = PK\{\alpha, \gamma | C_A = g^{\propto} h^{\gamma} \cap \propto \odot\ \mathrm{b}\}$$

where $\odot \in \{<, \leq, >, \geq\}$. The Prover knows assignments for $(\alpha, \beta, \gamma, \delta)$.

The proof relies on comparing the bit decompositions of a and b. The Prover computes Pedersen commitments to the bit decompositions and then proves they are formed correctly. Then, the Prover compares each i bit prefix of a and b; the results of the comparisons are stored in helper commitments D_i. The Prover creates an Equality Proof to show that the D_i are computed correctly. The committed value in D_{n-1} is equal to $\{-1,0,1\}$ depending on the relationship between a and b. The Prover adds an auxiliary proof showing that the committed value in D_{n-1} is equal to the appropriate value given $\odot$.

The U-Prove Cryptographic Specification [UPCS] allows the Prover, during the token presentation protocol, to create a Pedersen Commitment and show that the committed value is the equal to a particular token attribute. The Prover MAY use this Pedersen Commitment as either C_A or C_B. The Issuance and Token Presentation protocols are unaffected by this extension. The Prover may choose to create a range proof after these two protocols complete.

The committed values in C_A and C_B MUST NOT be hashed. If any of these values are U-Prove token attributes, the attributes also MUST NOT be hashed.

The Range Proof protocol makes use of the following U-Prove Extensions: Set Membership Proof Extension [EXSM], Bit Decomposition Extension [EXBD], and Equality Proof Extension [EXEQ].

1.1 Notation

In addition to the notation defined in [UPCS], the following notation is used throughout the document. The range proof consists of many sub-protocols; local variables are omitted from this list unless they consistently appear with the same meaning/value.

a	Value to be compared to b, known only to Prover.
b	Value to be compared to a, MAY be known to Verifier.
C_A	Pedersen Commitment to a. lOnly Prover knowns opening.
C_B	Pedersen Commitment to b, or null if Verifier knows b.
$bIsKnown$	True if Verifier knows b.

$\odot, proofType$	A value in the set $\{<, \leq, >, \geq\}$ indicating the relationship between a and b that needs to be proven.
min	Minimum possible value for a and b.
max	Maximum possible value for a and b.
$\mathcal{M}$	An equality map, as defined in U-Prove Equality Proof Extension [EXEQ]. Range proofs require multiple different equality maps; this document uses local variable $\mathcal{M}$ to refer to a map.
$\bar{\bar{A}}_i$	The value of a DL Equation, as defined in U-Prove Equality Proof Extension [EXEQ]. Range proofs create multiple different equality proofs; this document uses local variable $\bar{\bar{A}}_i$ to refer to the DL Equation values.
$\bar{\bar{g}}_{i,j}$	The bases of a DL Equation, as defined in U-Prove Equality Proof Extension [EXEQ]. Range proofs create multiple different equality proofs; this document uses local variable $\bar{\bar{g}}_{i,j}$ to refer to the DL Equation bases.
$\bar{\bar{x}}_{i,j}$	The witnesses (exponents) for a DL Equation, as defined in U-Prove Equality Proof Extension [EXEQ]. Range proofs create multiple different equality proofs; this document uses local variable $\bar{\bar{x}}_{i,j}$ to refer to the DL Equation witnesses.
$\vec{a} = (a_0, r_0), (a_1, r_1), \dots, (a_{n-1}, r_{n-1})$	The opening information for Pedersen Commitments $\vec{A}$. The a_i contain the bit decomposition of $a - min$, while the r_i are the second exponent.
$\vec{b} = (b_0, s_0), (b_1, s_1) \dots, (b_{n-1}, s_{n-1})$	The opening information for Pedersen Commitments $\vec{B}$. The b_i contain the bit decomposition of $b - min$, while the s_i are the second exponent. If the Verifier knows b, then $s_i = 0$.
$\vec{c} = (c_0, y_0), (c_1, y_1) \dots, (c_{n-1}, y_{n-1})$	The opening information for Pedersen Commitments $\vec{C}$. The c_i contain the difference between $\vec{a}$ and $\vec{b}$: $c_i = a_i - b_i$, while the y_i are the second exponent.
$\vec{d} = (d_1, t_1) \dots, (d_{n-1}, t_{n-1})$	The opening information for Pedersen Commitments $\vec{D}$. Each d_i stores the inequality relationship between the i least significant bits of a and b, represented as a value in $\{-1,0,1\}$. The t_i are the second exponent.
$\vec{e} = (e_1, v_1) \dots, (e_{n-1}, v_{n-1})$	The opening information for Pedersen Commitments $\vec{E}$. Each e_i is actually equal to d_{i-1}, while the v_i are the second exponent.
$\vec{x} = (c_1, m_1) \dots, (c_{n-1}, m_{n-1})$	The opening information for Pedersen Commitments $\vec{X}$. Each c_i is actually equal to the c_i in $\vec{c}$, while the m_i are the second exponent.
$\vec{A}$=$A_0, A_1, \dots, A_{n-1}$	Pedersen Commitments to $\vec{a}$.
$\vec{B} = B_0, B_1, \dots, B_{n-1}$	Pedersen Commitments to $\vec{b}$.
$\vec{C} = C_0, C_1, \dots, C_{n-1}$	Pedersen Commitments to $\vec{c}$.
$\vec{D} = D_1, \dots, D_{n-1}$	Pedersen Commitment to $\vec{d}$.
$\vec{E} = E_1, \dots, E_{n-1}$	Pedersen Commitment to $\vec{e}$.
$\vec{X} = X_1, \dots, X_{n-1}$	Pedersen Commitment to $\vec{x}$.
π_A	Proof that $\vec{A}$ is a valid commitment to the bit decomposition of $a - min$.
π_B	Proof that $\vec{B}$ is a valid commitment to the bit decomposition of $b - min$. Null if the Verifier knows b.

π_C Main equality proof showing that $\vec{D}$ and $\vec{X}$ are formed correctly.

π_D Auxiliary proof showing that D_{n-1} contains the correct value; either and equality proof or a set membership proof.

$a \leftarrow A$ Choose a uniformly at random from set A.

The key words "MUST", "MUST NOT", "SHOULD", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119].

1.2 Feature overview

The Prover knows the opening of a Pedersen Commitments $C_A = g^a h^r$ and $C_B = g^b h^s$ (optionally, b may be public knowledge). The Prover needs to show that the relationship $a \odot b$ holds, where $\odot \in \{<, \leq, >, \geq\}$ is also known to the Verifier. For efficiency, the Prover and Verifier both know that a and b fall inside the range $[min, max]$. The Prover will create a special-honest verifier zero-knowledge proof of knowledge that the Prover knows a tuple of values (a, r, b, s) such that:

1 $C_A = g^a h^r$.
2 $C_B = g^b h^s$.
3 The relationship $a \odot b$ holds, where $\odot \in \{<, \leq, >, \geq\}$.

The range proof consists of the following components:

1. Pedersen commitments $A_0, A_1, \dots, A_{n-1}$ to the bit decomposition of $a - min$, as well as a Bit Decomposition Proof [EXBD] showing the A_i are constructed correctly.
2. (Optional) Pedersen commitments $B_0, B_1, \dots, B_{n-1}$ to the bit decomposition of $b - min$, as well as a Bit Decomposition Proof [EXBD] showing the B_i are constructed correctly.
3. Pedersen commitments $X_0, \dots, X_{n-1}$ to $c_i = (a_i - b_i)^2$. These are helper values
4. Pedersen commitments $D_1, \dots, D_{n-1}$ to $d_i \in \{-1,0,1\}$, which represents the inequality relationship between the i least significant bits of a and b. We compute it as follows:

$$d_i = \begin{cases} a_i - b_i & i = 0 \\ d_{i-1} - d_{i-1}(a_i - b_i)^2 + (a_i - b_i) & i > 0 \end{cases}$$

5. An Equality Proof [EXEQ] showing the X_i and D_i are formed correctly.
6. An auxiliary proof showing that D_{n-1} is a commitment to the appropriate value in $\{-1,0,1\}$ given the type of inequality relationship the Prover is trying to prove.

2 Protocol specification

As the range proof can be performed independently of the U-Prove token presentation protocols, the common parameters consist simply of the group G_q, two generators g and h, and a cryptographic function $\mathcal{H}$. The commitments C_A and C_B MAY be generated by the Prover.

The remaining parameters may be chosen by either the Prover or Verifier: The values min and max indicate the maximum span for secret values a and b. The variable $bIsKnown$ indicates whether the Verifier knows b. The $proofType$ indicates the inequality relationship between a and b that the Prover wishes to demonstrate.

2.1 Common Protocols

The main body of the range proof is an Equality Proof defined in the U-Prove Equality Proof Extension [EXEQ] EQProofParams() returns the common parameters for the main proof. It generates an equality map $\mathcal{M}$ and sets up the DL equations $\bar{\bar{A}}_i = \prod_{j=0}^{n_i-1} \bar{\bar{g}}_{i,j}^{\alpha_{i,j}}$ where the $\bar{\bar{A}}_i$ and $\bar{\bar{g}}_{i,j}$ are public values returned by this protocol, while the $\alpha_{i,j}$ are secret values known only to the Prover.

EQProofParams( )

Input

Parameters: $desc(G_q)$,$\text{UID}_{\mathcal{H}}, g, h$

Commitment to a/b: $\vec{C} = C_0, C_1, \dots, C_{n-1}$

Commitment to d: $\vec{D} = D_1, \dots, D_{n-1}$

Commitment to $(a/b)^2$: $\vec{X} = X_1, \dots, X_{n-1}$

Commitment to e: $\vec{E} = E_1, \dots, E_{n-1}$

Computation

$\mathcal{M} := \emptyset$

$eq := 0$

$// \ D_i = g^{\delta_i} \cdot h^{\tau_i}$

For $i :=$ 0 to $n-1$

$\mathcal{M}.Add((\text{"delta"}, i), (eq, 0))$

$\bar{\bar{A}}_{eq} := D_i$

$\bar{\bar{g}}_{eq,0} := g$

$\bar{\bar{g}}_{eq,1} := h$

$eq := eq + 1$

End

$// \ A_i/B_i = g^{\chi_i} \cdot h^{\zeta_i}$

For $i :=$ 1 to $n-1$

$\mathcal{M}.Add((\text{"chi"}, i), (eq, 0))$

$\bar{\bar{A}}_{eq} := C_i$

$\bar{\bar{g}}_{eq,0} := g$

$\bar{\bar{g}}_{eq,1} := h$

$eq := eq + 1$

End

$// \ X_i = (A_i/B_i)^{\chi_i} \cdot h^{\mu_i}$

For $i :=$ 1 to $n-1$

$\mathcal{M}.Add((\text{"chi"}, i), (eq, 0))$

$\bar{\bar{A}}_{eq} := X_i$

$\bar{\bar{g}}_{eq,0} := C_i$

$1 := h$

$eq := eq + 1$

End

$// \ E_i = (X_i^{-1})^{\delta_{i-1}} \cdot h^{\nu_i}$

For $i :=$ 0 to $n-1$

$\mathcal{M}.Add((\text{"delta"}, i-1), (eq, 0))$

$\bar{\bar{A}}_{eq} := E_i$

$\bar{\bar{g}}_{eq,0} := X_i^{-1}$

$\bar{\bar{g}}_{eq,1} := h$

$eq := eq + 1$

End

Output

Return $\mathcal{M}, \bar{\bar{A}}, \bar{\bar{g}}$

Figure 1: EQProofParams.

2.2 Presentation

The Prover calls RangeProve to generate a range proof. We break up the range proof presentation protocol into various sub-protocols for ease of exposition. The range proof also requires calling protocols from Bit Decomposition Proof [EXBD], Set Membership Proof [EXSM], and Equality Proof [EXEQ].

RangeProve( )

Input

Parameters: desc(G_q), $\text{UID}_{\mathcal{H}}$, $g, h, min, max, bIsKnown, b, proofType$
Commitment to a: C_A
Opening information to C_A: a, r
Commitment to b: C_B
Opening information to C_B: b, s

Computation

$\vec{A}, \vec{a}, \pi_A, \vec{B}, \vec{b}, \pi_B$
$\quad := \text{GetBitProofs}\big(\text{desc}(G_q), \text{UID}_{\mathcal{H}}, g, h, min, max, bIsKnown, C_A, a, r, C_B, b, s\big)$
$\vec{C}, \vec{c} := \text{ComputeC}(\vec{A}, \vec{a}, \vec{B}, \vec{b})$
$\vec{D}, \vec{d} := \text{ComputeD}(\text{desc}(G_q), g, h, \vec{C}, \vec{c})$
$\vec{X}, \vec{x} := \text{ComputeX}(\text{desc}(G_q), g, h, \vec{C}, \vec{c})$
$\vec{E}, \vec{e} := \text{ComputeE}(\text{desc}(G_q), g, h, \vec{C}, \vec{c}, \vec{D}, \vec{d}, \vec{X}, \vec{x})$
$\mathcal{M}, \bar{\bar{A}}, \bar{\bar{g}} := \text{EQProofParams}(desc\big(G_q\big), g, h, \vec{C}, \vec{D}, \vec{X}, \vec{E})$
$\pi_C := \text{MainProof}(\text{desc}(G_q), \text{UID}_{\mathcal{H}}, g, h, n, \mathcal{M}, \bar{\bar{A}}, \bar{\bar{g}}, \vec{c}, \vec{d}, \vec{x}, \vec{e})$

If $proofType$ **is** $>$ then
$\quad \pi_D := \text{EqualityOfDL}(\text{desc}\big(G_q\big), \text{UID}_{\mathcal{H}}, g, h, 1, D_{n-1}, (d_{n-1}, t_{n-1}))$
Else if $proofType$ **is** $<$ then
$\quad \pi_D := \text{EqualityOfDL}(\text{desc}\big(G_q\big), \text{UID}_{\mathcal{H}}, g, h, -1, D_{n-1}, (d_{n-1}, t_{n-1}))$
Else if $proofType$ **is** $\geq$ then
$\quad \pi_D := \text{SetMembershipProve}(\text{desc}\big(G_q\big), \text{UID}_{\mathcal{H}}, g, h, \{0,1\}, D_{n-1}, (d_{n-1}, t_{n-1}))$
Else
$\quad \pi_D := \text{SetMembershipProve}(\text{desc}\big(G_q\big), \text{UID}_{\mathcal{H}}, g, h, \{0,1\}, D_{n-1}, (d_{n-1}, t_{n-1}))$
End

If $bIsKnown$ then
$\quad \vec{B} := \emptyset$
End

Output

Return $\vec{A}, \vec{B}, \vec{D}, \vec{X}, \pi_A, \pi_B, \pi_C, \pi_D$

Figure 2: RangeProve

The range proof requires dividing the bit decomposition of A by the bit decomposition of B to get an array of Pedersen commitments $\vec{C}$ and their openings $\vec{c}$. This step is performed in the function ComputeC().

ComputeC( )

Input

Parameters: $desc(G_q)$

Commitment to a: $\vec{A} = A_0, A_1, \ldots, A_{n-1}$

Opening information to A_i: $\vec{a} = (a_0, r_0), (a_1, r_1), \ldots, (a_{n-1}, r_{n-1})$

Commitment to b: $\vec{B} = B_0, B_1, \ldots, B_{n-1}$

Opening information to B_i: $\vec{b} = (b_0, s_0), (b_1, s_1) \ldots, (b_{n-1}, s_{n-1})$

Computation

For $i = 0$ **to** $n - 1$

$c_i \coloneqq a_i - b_i$

$y_i \coloneqq r_i - s_i$

$C_i \coloneqq A_i / B_i$

End

$\vec{C} \coloneqq C_0, C_1, \ldots, C_{n-1}$

$\vec{c} \coloneqq (c_0, y_0), (c, y_1) \ldots, (c, y_{n-1})$

Output

Return $\vec{C}, \vec{c}$

Figure 3: ComputeC

The range proof performs bit decompositions of a and b with the help of protocols from U-Prove Bit Decomposition Extension [EXBD]. For efficiency, it normalizes the range from $[min, max]$ to $[0, max - min]$. This step is important since the length of the range proof depends on the length of the bit decomposition. If the value of b is known to the Verifier, the Prover will generate a default Pedersen Commitments to the bit decomposition of b and omit the bit decomposition proof.

GetBitProofs( )

Input

Parameters: desc(G_q), $\mathrm{UID}_{\mathcal{H}}$, $g, h, min, max, bIsKnown$,
Commitment to a: C_A
Opening information to C_A: a, r
Commitment to b: C_B
Opening information to C_B: b, s

Computation

$n \coloneqq \lceil \log_2(max - min) \rceil$
$\tilde{a} \coloneqq a - min$
$\tilde{C}_A \coloneqq C_A \cdot g^{-min}$
$\vec{A}, \vec{a} \coloneqq \mathrm{GenerateBitDecomposition}(desc(G_q), g, h, n, \tilde{C}_A, \tilde{a}, r)$
$\pi_A \coloneqq \mathrm{BitDecompositionProve}(desc(G_q), \mathrm{UID}_{\mathcal{H}}, g, h, \tilde{C}_A, \vec{A}, \vec{a})$

$\tilde{b} \coloneqq b - min$
If $bIsKnown$ **then**
$\tilde{C}_B \coloneqq g^{\tilde{b}}$
$\vec{B}, \vec{b} \coloneqq \mathrm{DefaultBitDecomposition}(desc(G_q), g, h, n, \tilde{b})$
$\pi_B \coloneqq \emptyset$
Else
$\tilde{C}_B \coloneqq C_B \cdot g^{-min}$
$\vec{B}, \vec{b} \coloneqq \mathrm{GenerateBitDecomposition}(desc(G_q), g, h, n, \tilde{C}_B, \tilde{b}, s)$
$\pi_B \coloneqq \mathrm{BitDecompositionProve}(desc(G_q), \mathrm{UID}_{\mathcal{H}}, g, h, \tilde{C}_B, \vec{B}, \vec{b})$
End

Output

Return $\vec{A}, \vec{a}, \pi_A, \vec{B}, \vec{b}, \pi_B$

Figure 4: GetBitProofs.

The following two protocols generate a bit decomposition of an integer x and return Pedersen Commitments and their openings to this decomposition. GenerateBitDecomposition() generates random Pedersen Commitments, while DefaultBitDecomposition() sets the second exponent to 0.

GenerateBitDecomposition( )

Input

Parameters: $desc(G_q), g, h, n$
Commitment to x: C
Opening information to C: x, y

Computation

$x_0, x_1, \dots, x_{n-1} \leftarrow$ bit decomposition of x
$y_0, y, \dots, y_{n-1} \leftarrow \mathbb{Z}_q^*$
For $i :=\ 0$ **to** $n-1$
$C_i \coloneqq g^{x_i} h^{y_i}$
End
$\vec{C} \coloneqq C_0, C_1, \dots, C_{n-1}$
$\vec{x} \coloneqq (x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1})$

Output

Return $\vec{C}, \vec{x}$

Figure 5: GenerateBitDecomposition

DefaultBitDecomposition( )

Input

Parameters: $desc(G_q), g, h, n$
Integer: x

Computation

$x_0, x_1, \dots, x_{n-1} \leftarrow$ bit decomposition of x
$y_0, y, \dots, y_{n-1} \coloneqq 0{,}0, \dots, 0$
For $i :=\ 0$ **to** $n-1$
$C_i \coloneqq g^{x_i}$
End
$\vec{C} \coloneqq C_0, C_1, \dots, C_{n-1}$
$\vec{x} \coloneqq (x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1})$

Output

Return $\vec{C}, \vec{x}$

Figure 6: DefaultBitDecomposition

The range proof compares A to B bit by bit. It does so by computing Pedersen commitments $D_1, \dots, D_{n-1}$ to $d_i \in \{-1,0,1\}$, which represents the inequality relationship between the i least significant bits of a and b. We compute the d_i as follows:

$$d_i = \begin{cases} a_i - b_i & i = 0 \\ d_{i-1} - d_{i-1}(a_i - b_i)^2 + (a_i - b_i) & i > 0 \end{cases}$$

The function ComputeD() takes as input $c_i = a_i - b_i$, which is substituted into the above formula.

```
ComputeD( )
```

Input

Parameters: $desc(G_q), g, h$
Commitment to a/b: $\vec{C} = C_0, C_1, \dots, C_{n-1}$
Opening information to C_i: $\vec{c} = (c_0, y_0), (c_1, y_1) \dots, (c_{n-1}, y_{n-1})$

Computation

$d_0 \coloneqq c_0$
For $i \coloneqq 1$ **to** $n-1$
$d_i \coloneqq d_{i-1} - d_{i-1}{c_i}^2 + c_i$
$t_i \leftarrow \mathbb{Z}_q^*$
$D_i \coloneqq g^{d_i} h^{t_i}$
End
$\vec{D} \coloneqq D_1, \dots, D_{n-1}$
$\vec{d} \coloneqq (d_1, t_1), \dots, (d_{n-1}, t_{n-1})$

Output

Return $\vec{D}, \vec{d}$

Figure 7: ComputeD.

Proving that the D_i are formed correctly requires helper values $X_i = C_i^{c_i} h^{m_i}$.

```
ComputeX( )
```

Input

Parameters: $desc(G_q), g, h$
Commitment to a/b: $\vec{C} = C_0, C_1, \dots, C_{n-1}$
Opening information to C_i: $\vec{c} = (c_0, y_0), (c_1, y_1) \dots, (c_{n-1}, y_{n-1})$

Computation

For $i \coloneqq 1$ **to** $n-1$
$m_i \leftarrow \mathbb{Z}_q^*$
$X_i \coloneqq C_i^{c_i} h^{m_i}$
End
$\vec{X} \coloneqq X_1, \dots, X_{n-1}$
$\vec{x} \coloneqq (x_1, m_1), \dots, (x_{n-1}, m_{n-1})$

Output

Return $\vec{X}, \vec{x}$

Figure 8: ComputeX.

Proving that the D_i are formed correctly also requires helper values $E_i = (X_i^{-1})^{d_{i-1}} h^{\nu_i} = D_i \cdot (D_{i-1})^{-1} \cdot (C_i)^{-1}$.

ComputeE( )

Input

Parameters: $desc(G_q), g, h$

Commitment to a/b: $\vec{C} = C_0, C_1, \dots, C_{n-1}$

Opening information to $\vec{C}$: $\vec{c} = (c_0, y_0), (c_1, y_1) \dots, (c_{n-1}, y_{n-1})$

Commitment to d: $\vec{D} = D_1, \dots, D_{n-1}$

Opening information to $\vec{D}$: $\vec{d} = (d_1, t_1) \dots, (d_{n-1}, t_{n-1})$

Commitment to $(a/b)^2$: $\vec{X} = X_1, \dots, X_{n-1}$

Opening information to $\vec{X}$: $\vec{x} = (c_1, m_1) \dots, (c_{n-1}, m_{n-1})$

Computation

For $i :=$ 1 to $n - 1$

$\nu_i \coloneqq t_i - t_{i-1} + y_i + (d_{i-1} \cdot y_i \cdot c_i) + (d_{i-1} \cdot m_i)$

$E_i \coloneqq (X_i^{-1})^{d_{i-1}} h^{\nu_i}$

End

$\vec{E} \coloneqq E_1, \dots, E_{n-1}$

$\vec{e} \coloneqq (e_1, \nu_1), \dots, (e_{n-1}, \nu_{n-1})$

Output

Return $\vec{E}, \vec{e}$

Figure 9: ComputeE

The main body of the range proof is an Equality Proof [EXEQ] showing that $\vec{D}, \vec{X}, \vec{E}$ are formed correctly.

MainProof( )

Input

Parameters: $desc(G_q), \mathrm{UID}_{\mathcal{H}}, g, h,$

EQ Proof parameters: $\mathcal{M}, \bar{\bar{A}}, \bar{\bar{g}}$

Opening information to $\vec{C}$: $\vec{c} = (c_0, y_0), (c_1, y_1) \dots, (c_{n-1}, y_{n-1})$

Opening information to $\vec{D}$: $\vec{d} = (d_1, t_1) \dots, (d_{n-1}, t_{n-1})$

Opening information to $\vec{X}$: $\vec{x} = (c_1, m_1) \dots, (c_{n-1}, m_{n-1})$

Opening information to $\vec{E}$: $\vec{e} \coloneqq (e_1, v_1), \dots, (e_{n-1}, v_{n-1})$

Computation

$\bar{\bar{x}} \coloneqq \emptyset$

$eq \coloneqq 0$

$// \ D_i = g^{\delta_i} \cdot h^{\tau_i}$

For $i \coloneqq 0$ to $n-1$

$\bar{\bar{x}}_{eq,0} \coloneqq d_i$

$\bar{\bar{x}}_{eq,1} \coloneqq t_i$

$eq \coloneqq eq + 1$

End

$// \ A_i/B_i = g^{\chi_i} \cdot h^{\zeta_i}$

For $i \coloneqq 1$ to $n-1$

$\bar{\bar{x}}_{eq,0} \coloneqq c_i$

$\bar{\bar{x}}_{eq,1} \coloneqq y_i$

$eq \coloneqq eq + 1$

End

$// \ X_i = (A_i/B_i)^{\chi_i} \cdot h^{\mu_i}$

For $i \coloneqq 1$ to $n-1$

$\bar{\bar{x}}_{eq,0} \coloneqq c_i$

$\bar{\bar{x}}_{eq,1} \coloneqq m_i$

$eq \coloneqq eq + 1$

End

$// \ E_i = (X_i^{-1})^{\delta_{i-1}} \cdot h^{\nu_i}$

For $i \coloneqq 0$ to $n-1$

$\bar{\bar{x}}_{eq,0} \coloneqq e_i$

$\bar{\bar{x}}_{eq,1} \coloneqq v_i$

$eq \coloneqq eq + 1$

End

$\pi_C \coloneqq \mathrm{EqualityProve}(desc(G_q), \mathrm{UID}_{\mathcal{H}}, \bar{\bar{A}}, \bar{\bar{g}}, \mathcal{M}, \bar{\bar{x}})$

Output

Return π_C

Figure 10: MainProof.

EqualityOfDL is a small helper proof that shows that $D = g^d \cdot h^t$ is a Pedersen Commitment to some integer x known to the Verifier. The protocol generates an Equality Proof [EXEQ].

EqualityOfDL( )

Input

Parameters: $desc(G_q)$,$\mathrm{UID}_{\mathcal{H}}, g, h, x$
Commitment to d: D
Opening information to D: (d, t)

Computation

$\mathcal{M} := \emptyset$
$\bar{\bar{A}}_0 = D \cdot g^{-x}$
$\bar{\bar{g}}_{0,0} := h$
$\bar{\bar{x}}_{0,0} := t$
$\pi := \mathrm{EqualityProve}(desc(G_q),\mathrm{UID}_{\mathcal{H}}, \bar{\bar{A}}, \bar{\bar{g}}, \mathcal{M}, \bar{\bar{x}})$

Output

Return π

Figure 11: EqualityOfDL.

2.3 Verification

The Verifier receives the common parameters, as well as commitments to a and b and the proof. The Verifier returns true if the verification passes, false otherwise. Verification requires checking the bit decomposition proofs π_A and π_B, the main equality proof π_C, and the auxiliary proof π_D that depends on the proof type.

RangeVerify( )

Input

Parameters: desc(G_q), $\text{UID}_{\mathcal{H}}$, $g, h, min, max, bIsKnown, b, proofType$
Commitment to a: C_A
Commitment to b: C_B
Proof: $\vec{A}, \vec{B}, \vec{D}, \vec{X}, \pi_A, \pi_B, \pi_C, \pi_D$

Computation

$P \coloneqq true$
$P := P$ **AND** $\text{BitDecompositionVerify}(\text{desc}(G_q), \text{UID}_{\mathcal{H}}, g, h, C_A/g^{min}, \vec{A}, \pi_A)$
If $bIsKnown$ then
$\vec{B}, \vec{b} \coloneqq \text{DefaultBitDecomposition}(desc(G_q), g, h, n, b - min)$
Else
$P := P$ **AND** $\text{BitDecompositionVerify}(\text{desc}(G_q), \text{UID}_{\mathcal{H}}, g, h, C_B/g^{min}, \vec{B}, \pi_B)$
End

$\vec{C} \coloneqq \text{ComputeClosedC}(\text{desc}(G_q), \vec{A}, \vec{B})$
$\vec{E} \coloneqq \text{ComputeClosedE}(\text{desc}(G_q), \vec{D}, \vec{C})$
$\mathcal{M}, \bar{\bar{A}}, \bar{\bar{g}} \coloneqq \text{EQProofParams}(desc(G_q), g, h, \vec{C}, \vec{D}, \vec{X}, \vec{E})$
$P \coloneqq P$ **AND** $\text{EqualityVerify}(\text{desc}(G_q), \text{UID}_{\mathcal{H}}, \bar{\bar{A}}, \bar{\bar{g}}, \mathcal{M}, \pi_C)$

If $proofType$ **is** $>$ then
$P \coloneqq P$ **AND** $\text{EqualityOfDLVerify}(\text{desc}(G_q), \text{UID}_{\mathcal{H}}, g, h, D_{n-1}, 1, \pi_D)$
Else if $proofType$ **is** $<$ then
$P \coloneqq P$ **AND** $\text{EqualityOfDLVerify}(\text{desc}(G_q), \text{UID}_{\mathcal{H}}, g, h, D_{n-1}, -1, \pi_D)$
Else if $proofType$ **is** $\geq$ then
$P \coloneqq P$ **AND** $\text{SetMembershipVerify}(\text{desc}(G_q), \text{UID}_{\mathcal{H}}, g, h, D_{n-1}, \{0,1\}, \pi_D)$
Else
$P \coloneqq P$ **AND** $\text{SetMembershipProve}(\text{desc}(G_q), \text{UID}_{\mathcal{H}}, g, h, D_{n-1}, \{-1,0\}, \pi_D)$
End

Output

Return P

The Verifier uses the function ComputeClosedC() to compute $C_i = A_i/B_i$, which are needed to verify π_C.

ComputeClosedC( )

Input

Parameters: $desc(G_q)$

Commitment to a: $\vec{A} = A_0, A_1, \dots, A_{n-1}$

Commitment to b: $\vec{B} = B_0, B_1, \dots, B_{n-1}$

Computation

For $i :=\ 0$ **to** $n-1$

$C_i \coloneqq A_i / B_i$

End

$\vec{C} \coloneqq C_0, C_1, \dots, C_{n-1}$

Output

Return $\vec{C}$

The Verifier calls function ComputeClosedE() to compute $E_i = D_i \cdot (D_{i-1})^{-1} \cdot C_i^{-1}$, which are needed to verify π_C.

ComputeClosedE( )

Input

Parameters: $desc(G_q)$

Commitment to d: $\vec{D} = D_1, \dots, D_{n-1}$

Commitment to b: $\vec{C} = C_0, C_1, \dots, C_{n-1}$

Computation

$D_0 \coloneqq C_0$

For $i :=\ 1$ **to** $n-1$

$E_i \coloneqq D_i \cdot (D_{i-1})^{-1} \cdot C_i^{-1}$

End

$\vec{E} \coloneqq E_0, E_1, \dots, E_{n-1}$

Output

Return $\vec{E}$

The Verifier calls EqualityOfDLVerify to check that D is a Pedersen Commitment to x.

EqualityOfDLVerify( )

Input

Parameters: $desc(G_q)$,$\mathrm{UID}_{\mathcal{H}}, g, h, x$
Commitment to d: D
Proof: π

Computation

$\mathcal{M} := \emptyset$
$\bar{\bar{A}}_0 := D \cdot g^{-x}$
$\bar{\bar{g}}_{0,0} := h$
$pass := \mathrm{EqualityVerify}(desc(G_q),\mathrm{UID}_{\mathcal{H}}, \bar{\bar{A}}, \bar{\bar{g}}, \mathcal{M}, \pi)$

Output

Return $pass$

3 Security Considerations

The range proof invokes protocols from U-Prove Equality Proof Extension [EXEQ], U-Prove Bit Decomposition Extension [EXBD], and U-Prove Set Membership Proof Extension [EXSM]. Its security relies on their security. The following restriction apply:

- The Prover and the Verifier MUST NOT know the relative discrete logarithm $\log_g h$ of the generators g and h. This is not an issue if the generators are chosen from the list of U-Prove recommended parameters.

References

[Brands] Stefan Brands. *Rethinking Public Key Infrastructures and Digital Certificates.* The MIT Press, August 2000. http://www.credentica.com/the_mit_pressbook.html.

[EXBD] Mira Belenkiy. *U-Prove Bit Decomposition Extension*. Microsoft, June 2014. http://www.microsoft.com/u-prove.

[EXEQ] Mira Belenkiy. *U-Prove Equality Proof Extension*. Microsoft, June 2014. http://www.microsoft.com/u-prove.

[EXSM] Mira Belenkiy. *U-Prove Set Membership Proof Extension*. Microsoft, June 2014. http://www.microsoft.com/u-prove.

[RFC2119] Scott Bradner. *RFC 2119: Key words for use in RFCs to Indicate Requirement Levels*, 1997. ftp://ftp.rfc-editor.org/in-notes/rfc2119.txt.

[UPCS] Christian Paquin, Greg Zaverucha. *U-Prove Cryptographic Specification V1.1 (Revision 3).* Microsoft, December 2013. http://www.microsoft.com/u-prove.